A Mobile-Agent Approach for Location Tracking in a Wireless Sensor Network

Yu-Chee Tseng, Sheng-Po Kuo, Hung-Wei Lee, and Chi-Fu Huang

Department of Computer Science and Information Engineering
National Chiao-Tung University
Hsin-Chu, 30050, Taiwan
E-mail: {yctseng, spkuo, leehw, and, cfhuang}@csie.nctu.edu.tw
(corresponding author: Prof. Yu-Chee Tseng)

Abstract

The wireless sensor network is an emerging technology that may greatly facilitate human life by providing ubiquitous sensing, computing, and communication capability, through which people can more closely interact with the environment wherever he/she goes. To be context-aware, one of the central issues in sensor networks is *location tracking*, whose goal is to monitor the roaming path of a moving object. While similar to the location-update problem in PCS networks, this problem is more challenging since there are no central control mechanism and backbone network in such environment. In this paper, we propose a novel protocol based on the *mobile agent* paradigm. Once a new object is sensed, a mobile agent will be initiated to track the roaming path of the object. The agent is mobile since it will choose the sensor closest to the object to stay. The agent may invite some nearby slave sensors to cooperatively position the object and inhibit other irrelevant (i.e., farther) sensors from tracking the object. As a result, the communication and sensing overheads are greatly reduced. Our prototyping of the location-tracking mobile agent based on IEEE 802.11b NICs and our experimental experiences are also reported.

Keywords: ad hoc network, context-aware computing, location tracking, mobile computing, sensor network, wireless communication.

1 Introduction

The rapid progress of wireless communication and embedded micro-sensing MEMS technologies have made *wireless sensor networks* possible. Such environments may have many inexpensive wireless nodes, each capable of collecting, processing, and storing environmental information, and communicating with neighboring nodes. In the past, sensors are connected by wire lines. Today, this environment is combined with the novel *ad hoc* networking technology to facilitate inter-sensor communication [7]. The flexibility of installation and configuration is greatly improved. A flurry of research activities have recently been commenced in sensor networks.

With sensor networks, human can interact with environments more closely. Grouping thousands of sensors together may revolutionize information gathering. For example, a disaster detector may be set up so that temperatures of a forest can be monitored by sensors to prevent small harmless brush fires from becoming monstrous infernos. Similar techniques can be applied to flood and typhoon detection. Another application is environment control; sensors can monitor factors such as temperature and humidity and feed these information back to a central air conditioning and ventilation system. By attaching sensors on vehicles, roads, traffic lights, traffic information can be fed to a central office in a real-time manner, which can be used to smartly control traffic lights. Even more, vehicles may exchange on-line traffic information as they pass each other. Sensors may also be used in combination with GPS to improve positioning accuracy. However, many issues remain to be resolved for the success of sensor networks.

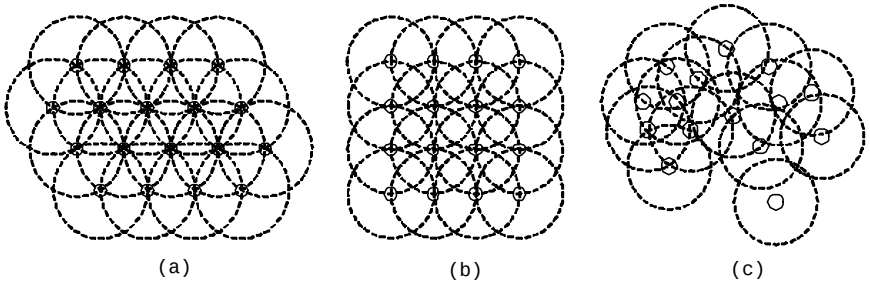

Figure 1: (a) Triangular, (b) square, and (c) irregular sensor networks.

- ***Scalability:*** Since a sensor network typically comprises a large number of nodes, how to manage these huge number of nodes by balancing related performance issues is not an easy job. Distributed and localized algorithms are essential in such environments [1, 5, 6]. Also, scalability is a critical issue in handling the large number of communications. In [11, 12], the *coverage* and *exposure* of an irregular sensor network are discussed by utilizing computational geometry and graph techniques. This coverage problem can be considered as the Art Gallery Problem. The Art Gallery Problem was solved optimally in 2D [9] and was shown to be NP-hard in 3D case. The coverage formulation can be used to solve some location-related problems [13]. Regular placement of sensors and their sensing ability are discussed in [3] and [10].

- ***Stability:*** Since sensors are likely to be installed in outdoor or even hostile environments, it is reasonable to assume that device failures would be regular and common events. Protocols should be stabile and fault-tolerant. Time synchronization among sensors is discussed in [2].

- ***Power-saving:*** Since no plug-in power is available, sensor devices will be operated by battery powers. Energy conservation should be kept in mind in all cases. Energy consumption of communications might be a major factor. Techniques such as data fusion may be necessary [2], but the timeliness of data should be considered too. Data dissemination is investigated in [4]. Mobile agent-based solutions are sometimes more power-efficient [8].

Since one of the goals of sensor networks is to monitor the environment, one fundamental issue is the *location-tracking problem*, whose goal is to trace the roaming paths of moving objects in the network area. This problem is similar to the location-update problem in PCS networks, but is more challenging since there are no central control mechanism and backbone network in such environment. In this paper, we propose a novel protocol based on the *mobile agent* paradigm. Once a new object is sensed, a mobile agent will be initiated to track the roaming path of the object. The agent is mobile since it will choose the senor closest to the object to stay. That is, the agent actually follows the object by hopping from sensor to sensor. The agent may invite some nearby slave sensors to cooperatively position the object and inhibit other irrelevant (i.e., farther) sensors from tracking the object. As a result, the communication and sensing overheads are greatly reduced. Our prototyping of the location-tracking mobile agent based on IEEE 802.11b NICs and our experimental experiences are also reported.

The organization of this paper is as follows. Section 2 describes our network model and defines the location-tracking problem. Our protocol based on mobile agents is presented in Section 3. Our prototyping and experimental experience is given in Section 4. Section 5 draws on conclusions.

2 Network Model and Problem Statement

We consider a sensor network, which consists of a set of sensor nodes placed in a 2D plane. Sensors may be arranged as a regular or irregular network, as shown in Fig. 1. However, for ease of presentation, throughout this paper we will assume a triangular network as illustrated in Fig. 1(a), where each set of three neighboring sensors forms an

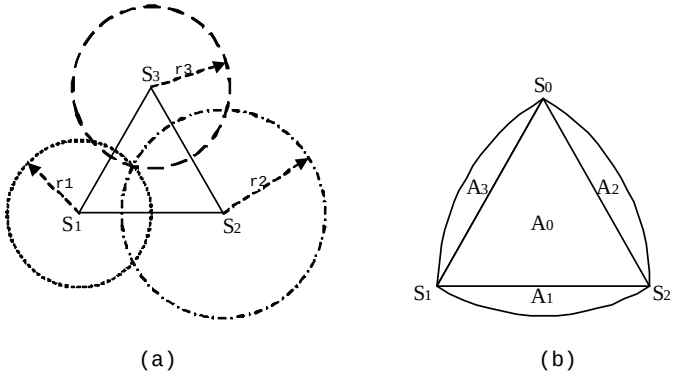

Figure 2: (a) Positioning example and (b) working area and backup areas.

equilateral triangle [1]. In order to track objects' locations, each sensor is aware of its physical location as well as the physical locations of its neighboring sensors. Each sensor has sensing capability as well as computing and communication capabilities, so as to execute protocols and exchange messages with neighboring sensors.

Each sensor is able to detect the existence of nearby moving objects. We assume that the sensing scope is r, which is equal to the side length of the triangles [2]. Within the detectable distance, a sensor is able to determine its distance to an object. This can be achieved either by the fly time or strength of the signals that are transmitted by the object or of the signals that are transmitted by the sensor and reflected from the object.

We assume that three sensors are sufficient to determine the location of an object. Specifically, suppose that an object resides within a triangle formed by three neighboring sensors S_1, S_2, and S_3. Suppose that the distances to the object detected by these sensors are r_1, r_2, and r_3, respectively. As shown in Fig. 2(a), by the intersections of the circles centered at S_1 and S_2, two possible positions of the object can be determined. With the assistance of S_3, the precise position can be determined. (It should be noted that in practice errors may exist, and thus more sensors will be needed to to improve the accuracy.)

The goal of this work is to determine the roaming path of a moving object in the sensor network. The trace of the object should be reported to an external server from time to time, depending on whether this is a real-time application or not. The intersection of the sensing scopes of three neighboring is as shown in Fig. 2(b). We further divide the area into one *working area* A_0 and three *backup areas* $A_1, A_2,$ and A_3. The working area defines the scope when these three sensors work normally, while the backup areas specify the "handover" region when objects move into other sensors' working areas.

3 The Location Tracking Protocol

3.1 Basic Idea

Our location-tracking protocol is derived by the cooperation of sensors. Whenever an object is detected, an *election process* will be conducted by some nearby sensors to choose a sensor, on which an agent will be initiated, to monitor the roaming behavior of the object. As the object moves, the agent may migrate to the sensor that is closer to the object to keep on monitoring the object. Fig. 3 illustrates this concept, where the dash line is the roaming path of the object, and arrows the migration path of the agent. By so doing, the computation and communication overheads can be reduced significantly.

[1] Although our development is based on regular networks, our framework should easily extend to more complicated regular networks or even irregular networks.

[2] In practice, r should be slightly larger than the side length. We make such an assumption for ease of presentation.

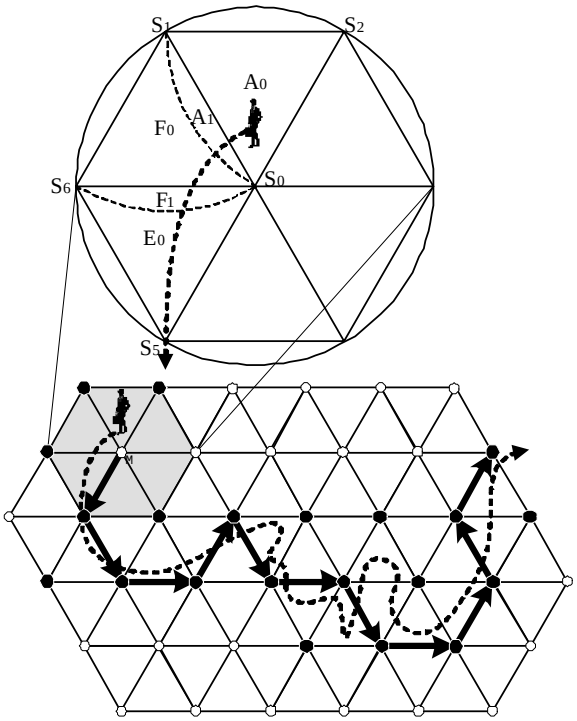

Figure 3: Roaming path of an object (dash line) and the migration path of the corresponding master agent (arrows). Sensors that ever host a slave agent are marked by black.

Recall that positioning an object requires the cooperation of at least three sensors. The mobile agent, or called the *master*, will invite two neighboring sensors to participate by dispatching a *slave* agent to each of them. These three agents (master and slaves) will cooperate to perform the triangle-positioning algorithm. From time to time, the slaves will report their positioning results to the master agent, who will then calculate the target's precise locations. As the object moves, these slave agents may be revoked and reassigned. Certain signal strength thresholds will be used to determine when to revoke/reassign a slave agent. The details will be given later. In Fig. 3, those sensors that ever host a slave agent are marked by black. We comment that although our development is based on the cooperation of two slave agents, it will be straightforward to extend our work to more slave agents to improve the positioning accuracy. To reduce the amount of data to be carried on, a master may decide to forward some histories to the backbone database. The time to carry out this job is application-dependent and is beyond the scope of this paper.

Observe the top part of Fig. 3. We now discuss how slave agents are revoked and reassigned. When resident in the working area A_0, the object is tracked by sensors S_0, S_1, and S_2. On entering the backup area A_1, since the signals received by S_2 will reduce to a level below a threshold, the slave agent at S_2 will be revoked and a new slave will be issued to S_6. Similarly, on entering the backup area F_1, the slave at S_1 will be revoked, and a new one will be issued to S_5. As the object passes S_5, the master itself will lose the target, in which case the master will migrate itself to S_5. All old slaves will be revoked and new slaves will be invited.

When an object is in the backup areas of some sensors, it is possible that it can be sensed by more than three sensors. To reduce the sensing overheads, master and slave agents can inhibit other irrelevant sensors from monitoring the object. This concept is illustrated in Fig. 4. The object is currently in area A_0. Sensors $S_3, S_4, \ldots, S_{11}$, which may sometimes detect the object, will be inhibited from tracking this object by warning signals that are issued periodically by the agents in $S_0, S_1,$ and S_2.

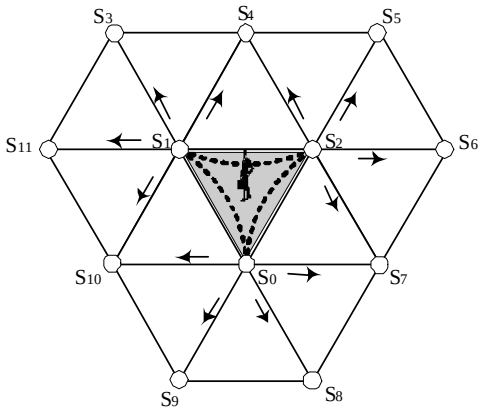

Figure 4: Inhibiting farther sensors $S_3, S_4, \ldots, S_{11}$ from monitoring the object.

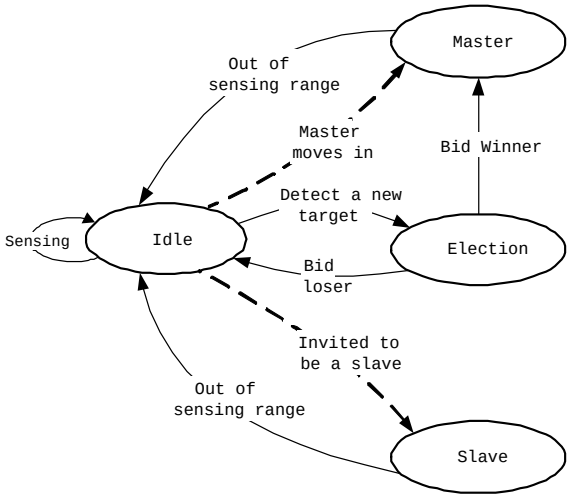

Figure 5: State transition diagram of a sensor (for one particular object).

3.2 Protocol Details

Below, we formally develop our tracking protocol. Since there may exist multiple objects in the network, we have to assume that it is possible for sensors to distinguish one object from the other. This can be done by having each object periodically send a unique ID code. Otherwise, some mechanism is needed for sensors to combine proper signals from proper sensors for different objects.

The following discussion will focus on one particular object. For each object, three or even more sensors will be able to detect its existence. Fig. 5 shows the state transition diagram of each sensor. (It should be noted that for different objects, a sensor may stay in different states.) Initially, each sensor is in the *idle* state and performs the *Basic Protocol*. Under this state, a sensor will continuously detect any object appearing in its sensing scope. Once detecting a new object, the sensor will enter the *election* state and perform the *Election Protocol* to bid for serving as a master. Most likely, the sensor that is closest to the object will win and become the master agent, which will then dispatch two slave agents to two nearby sensors. The master will go to the *master* state and perform the *Master Protocol*, while the slaves will go to the *slave* state and perform the *Slave Protocol*. To prevent too frequent moves of the agents, as long as the object remains in the working area, the states will not change. However, once the object enters the backup areas, the roles of master and slave may be changed. In this case, an idle sensor may be invited to serve as a master or slave. Another case that a sensor may stay in the idle state is when it detects an object

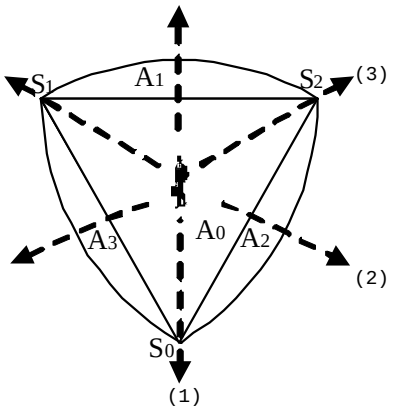

Figure 6: Possible roaming tracks for an object to leave a triangle.

in its backup areas and keeps on receiving inhibiting messages from neighboring sensors. This is reflected by the self-looping transition for the idle state.

Fig. 6 shows six tracks that an object may leave a triangle. Suppose that the master is currently in S_0, and the two slaves in S_1 and S_2. By the symmetry, these can be reduced 3 tracks (numbered by 1 to 3). For track 1, the master discovers two slaves losing the target simultaneously. So the master will revoke all slaves and reassign two new slaves. For track 2, only the slave agent in S_1 will be revoked, and a new one will be invited. For track 3, the master discovers one slave as well as itself losing the target. In this case, the master should migrate itself to the sensor that can still detect the object (typically with the strongest receive signals) and revoke all current slaves. After moving to the new sensor, two new slaves should be invited. Finally, we comment that the object may move too fast to be detected. If so, sensors may suddenly lose the target. As a last resort, all agents,when losing the object for a timeout period, will be dissolved. Since no inhibiting message will be heard, all sensors must remain in the idle state for this particular object, and new election process will take place to choose a new master to track this object. Our protocol is thus quite fault-tolerant in this sense.

Each sensor will keep an *object list (OL)* to record the status of all targets in its sensing scope. Each entry in OL is indexed by the object's unique identity, denoted by ID. For each object, there are two sub-fields: *status* and *time-stamp*. *ID.status* can be one of the four values: *Master*, *Slave*, *Standby*, and *Inhibited*. *ID.time-stamp* is the time when the record is last updated.

Seven types of control messages may be sent by our protocol.

(1) *bid_master(ID, sig):* This is for a sensor to compete as a master for object ID, if no inhibiting record has been created in OL for ID. The parameter sig reflects the receive signal strength for this object.

(2) *assign_slave(ID, s_i, t):* This is for a master to invite a nearby sensor s_i to serve as slave agent for object ID for an effective time interval of t.

(3) *revoke_slave(s_i):* This is for a master to revoke its slave at sensor s_i.

(4) *inhibit(ID):* This is a broadcast message for a master/slave to inhibit neighboring irrelevant sensors from tracking object ID. The effective time of the inhibiting message is defined by a system parameter T_{inh}.

(5) *release(ID):* This is to invalidate an earlier inhibiting message.

(6) *move_master(ID, s_i, $hist$):* A master uses this message to migrate itself from its current sensor to a nearby sensor s_i, where $hist$ carries all relevant codes/data/roaming histories related to object ID.

(7) *data(ID, sig, ts):* A slave uses this packet to report to its master the tracking results (sig =signal strength and ts = timestamp) for ID.

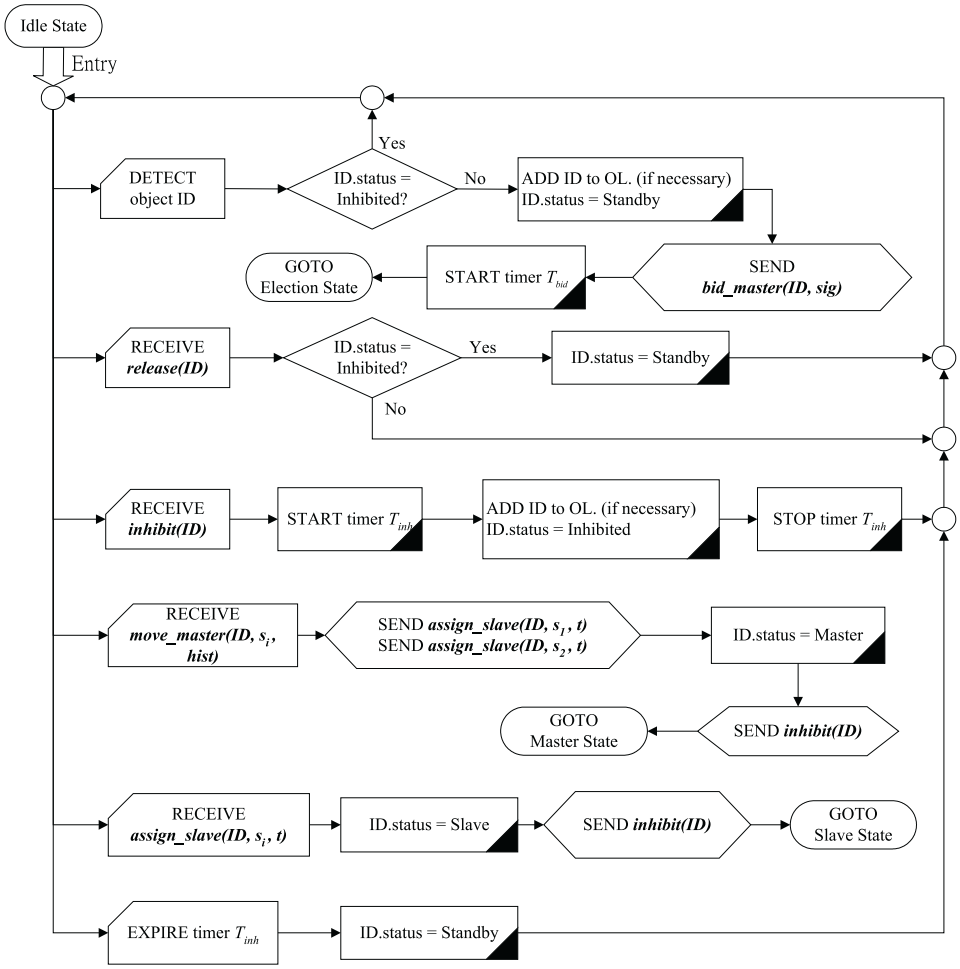

Figure 7: The Basic Protocol.

Below, we formally present our four protocols. The Basic Protocol is shown in Fig. 7. This is an endless loop containing 6 event-driven actions. The first one describes the reaction when detecting an object. If an inhibiting record exists, it will ignore the object. Otherwise, the sensor will go to the election state. The next four events describe the reactions when receiving a message from a neighboring sensor. In particular, if an *inhibit(ID)* message is received, a timer $T_{inh}(ID)$ will be set. The last one describes the reaction when the above timer expires, in which case the object's status will be changed to Standby and the sensor will be allowed to monitor this object.

The Election Protocol is shown in Fig. 8. In the beginning, a *bid_master* message will be sent and a timer $T_{bid}(ID)$ will be set. Then the sensor will wait for three possible events to occur: receiving *bid_master*, receiving *inhibit*, and finding timer T_{bid} expired. Signal strength will be used in the competition. Depending of different events, the sensor will go to the Master, or Idle state.

Fig. 9 shows the Master Protocol. The first event is to collect data from neighboring sensors. The next two events are for slave agents and the master agent itself losing the target,respectively. Note that the areas A1, A2, and A3 refer to Fig. 2(b). The last event is to inhibit irrelevant sensors from monitoring the object.

The Slave Protocol is shown in Fig. 10. The first event controls the timing, by timer T_{rep}, to report data to the master. The second event is to revoke the slave. The last event is to inhibit other irrelevant sensors.

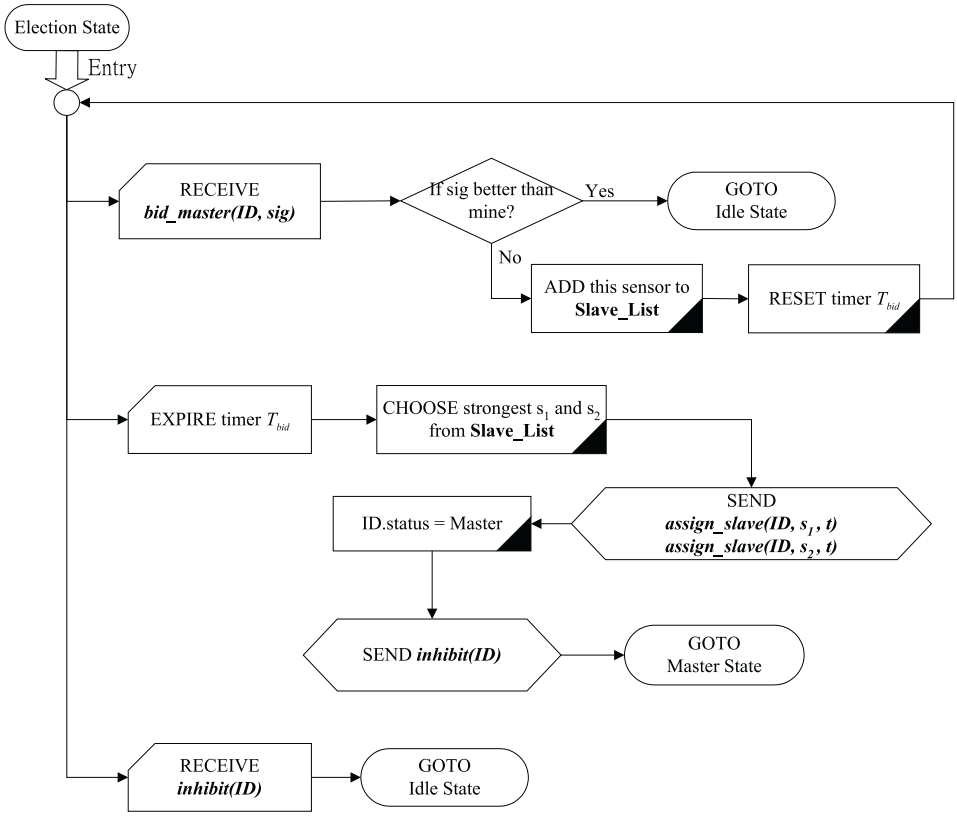

Figure 8: The Election Protocol.

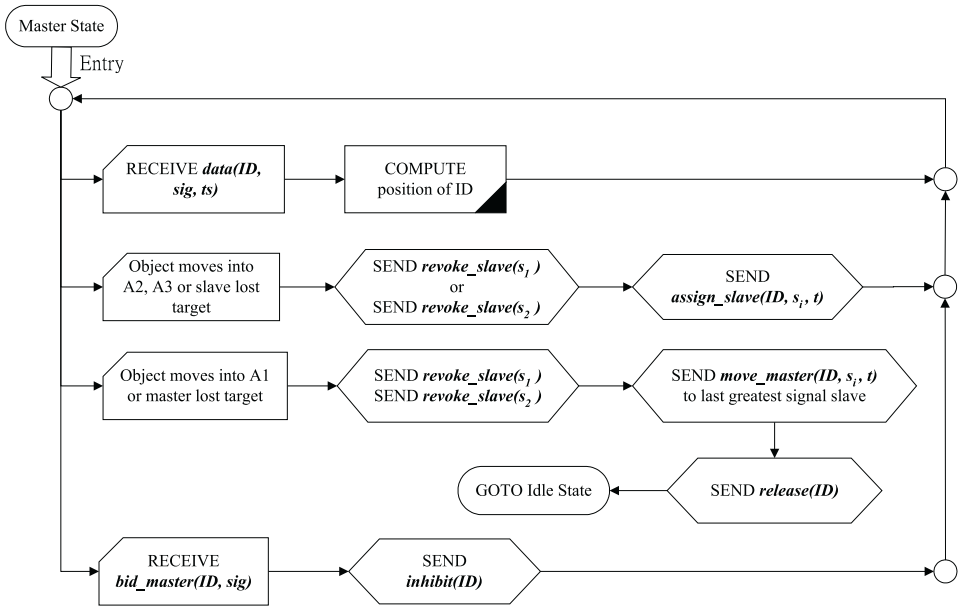

Figure 9: The Master Protocol.

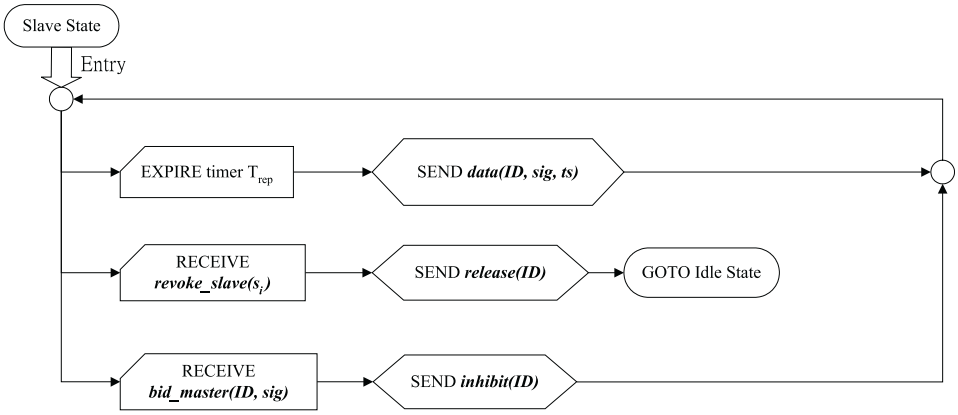

Figure 10: The Slave Protocol.

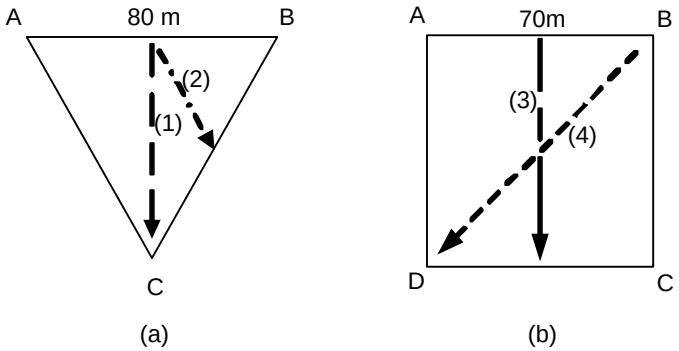

Figure 11: Experimental environment: (a) triangular sensor network and (b) square sensor network. Dash lines represented tested roaming paths.

4 Prototyping and Experimental Experience

In order to verify the feasibility of the proposed protocol, we have prototyped a system based on IEEE 802.11 NICs. Signal strength is used as the criterion to position objects. Specifically, four IBM laptops each equipped with a Lucent ORiNOCO 802.11b WaveLAN card are used. Three of them are placed as an equilateral triangle each separated by 80 meters to emulate sensor nodes, as shown in Fig. 11(a). One laptop is used to simulate the roaming object by periodically broadcasting beacons. For better sensitivity, an extra WaveLAN Range Extender Antenna is attached to this laptop. The sensor nodes monitor the received beacon strength transmitted by the object using the Client Manager utility.

First, we measure the degradation of signal strength versus distance. Fig. 12 shows one set of data that we collected. For every 5 meters from 0 to 100 meters a measurement is recorded. As can be expected, signal strengths received from IEEE 802.11b are not very stable. To resolve this problem, we use the "regression quadratic polynomial" to smooth out the curve, as illustrated by the solid line in Fig. 12. The curve is used to convert a received signal strength to an estimated distance.

Since signal strength is not an accurate measurement, the aforementioned triangle-positioning algorithm can not be applied directly. In fact, as one may expect, signal strengths change all the time, even under a motionless situation. Certain gaps inherently exist between estimated distances and actual distances. The real situation is as shown in Fig. 13, where the three estimated circles centered at the sensors have no common intersection.

As a result, we propose an approximation algorithm as follows. Let A, B, and C be the sensor nodes,which

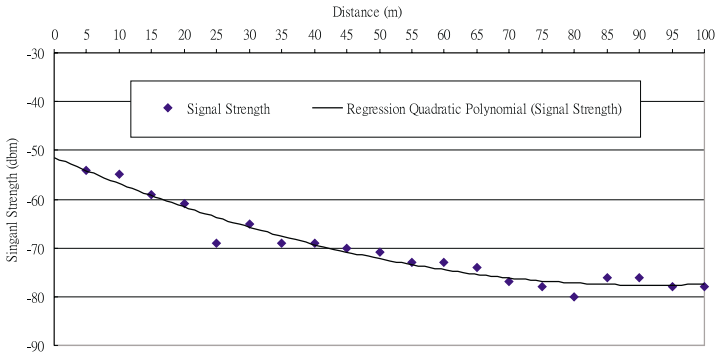

Figure 12: Experiment of signal strength vs. distance for IEEE 802.11b.

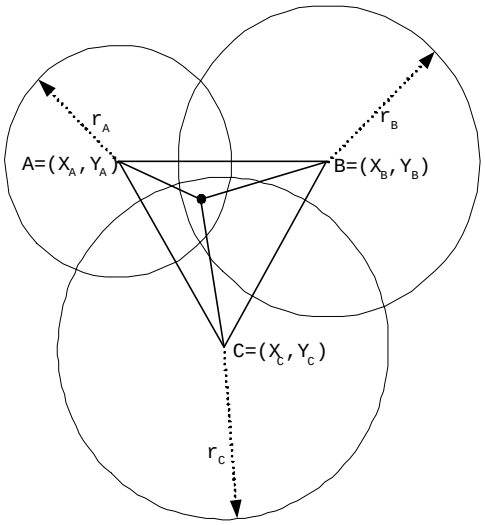

Figure 13: The position approximation algorithm.

are located at (x_A, y_A), (x_B, y_B), and (x_C, y_C), respectively. For any point (x, y) on the plane, we then define a difference function

$$\begin{aligned}\sigma_{x,y} = &|\sqrt{(x-x_A)^2+(y-y_A)^2} - r_A| \\ &+ |\sqrt{(x-x_B)^2+(y-y_B)^2} - r_B| \\ &+ |\sqrt{(x-x_C)^2+(y-y_C)^2} - r_C|,\end{aligned}$$

Where r_A, r_B, and r_C are the estimated distances to A, B, and C respectively. The location of the object is determined to be the point (x, y) among all points such that its difference function $\sigma_{x,y}$ is minimized. In our experiment, we consider only grid points on the plane that have integer coordinates in both x- and y-axes. We measure the location of the object every second. Furthermore, to take sudden fluctuation of signal strength into account, we enforce a condition that the object does not move faster than 5 meters per second. As a result, when searching for the object's location, only those points in $(x \pm 5, y \pm 5)$ are evaluated for their difference functions, where (x, y) represents the location in the previous measurement.

Our experiments were done in an outdoor, plain area where not much interference was expected. Two roaming paths as illustrated in Fig. 11(a) were tested. For roaming path (1), three sets of results are demonstrated in Fig. 14. For roaming path (2), the results are demonstrated in Fig. 15. As can be seen, the predicted paths are close to the actual roaming paths, but there are still large gaps yet remaining to be improved further.

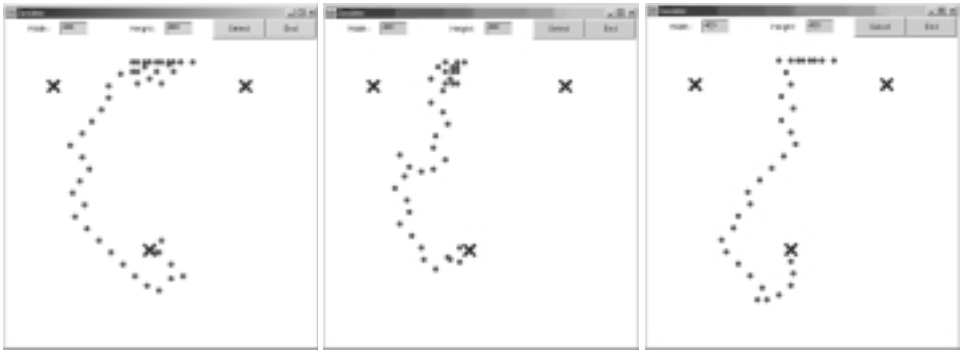

Figure 14: Tracking result of path (1) in Fig. 11(a).

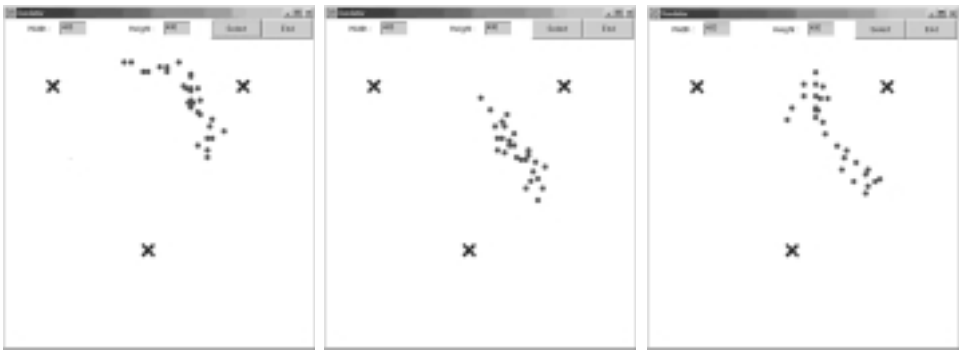

Figure 15: Tracking result of path (2) in Fig. 11(a).

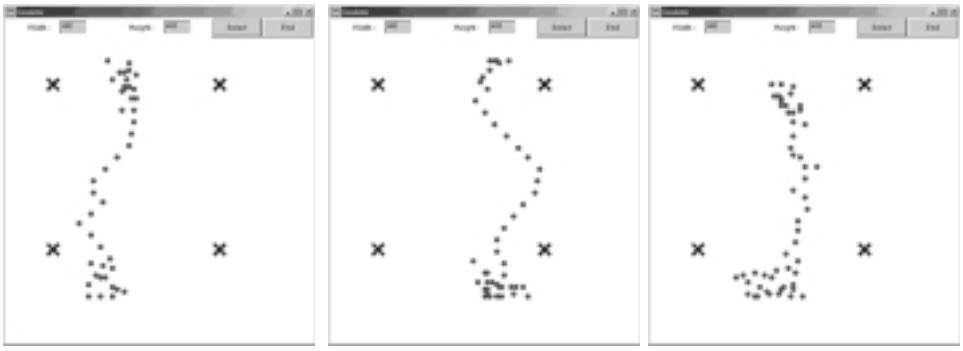

Figure 16: Tracking result of path (3) in Fig. 11(b).

Figure 17: Tracking result of path (4) in Fig. 11(b).

We have also tested the arrangement in Fig. 11(b), where four sensors arranged as a square are used. The extension for the tracking protocol and positioning algorithm is straightforward. Our tested results are shown in Fig. 16 and Fig. 17.

5 Conclusions and Future Work

We have proposed a novel location-tracking protocol that can be used for positioning moving objects in a sensor network with regular and irregular topologies. A mobile agent approach is adopted, and agents are able to roam around to follow the moving objects. Hence the communication and sensing overheads can be greatly reduced. We have prototyped our agents based on an environment with IEEE 802.11b wireless LAN cards. Signal strengths are used as the criterion to measure an object's position. While our protocol is proved to work correctly, the accuracy of using 802.11b NICs is unsatisfactory and remains to be improved further. Larger-scale experiments are being planned by using simulations based NS2 and the related results (such as tracking costs) are expected to be presented in our future work. Further, in more complex situations, such as terrains, buildings, and even indoor environments, the accuracy problem will be more serious and deserves further investigation.

References

[1] A. Savvides, C.C. Han, M.B. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Procs. of MobiCOM*, 2001.

[2] B. Horling, R. Vincent, R. Mailler, J. Shen, R. Becker, K. Rawlins, and V. Lesser. Distributed Sensor Network for Real Time Tracking. In *Proc. of the 5th international conference on Autonomous agents*, pages 417–424, 2001.

[3] B. Huang, W. Zhang, and Z. Guo. A study of spatial structures of sensor networks and multi-agent negotiation strategies, (this is an appendix to a quarterly report.), 2001. http://www.cs.wustl.edu/ zhang/projects/dcmp/.

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proc. of MobiCOM*, pages 56–67, 2000.

[5] C. Savarese, J. Rabaey, J. Beutel. Locationing in distributed ad-hoc wireless sensor networks. In *Proc. of the ICASSP*, 2001.

[6] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: scalable coordination in sensor networks. In *Proc. of MobiCOM*, pages 263–270, 1999.

[7] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[8] Hairong Qi, S.S. Iyengar, and K. Chakrabarty. Multiresolution data integration using mobile agents in distributed sensor networks. *IEEE Tran. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 31(3):383–391, 2001.

[9] J. O'Rourke. Computational geometry column 15. *International Journal of Computational Geometry and Applications*, 2(2):215–217, 1992.

[10] K. Chakrabarty, S.S. Iyengar, Hairong Qi, and Eungchun Cho. Coding theory framework for target location in distributed sensor networks. In *Proc. Intl. Symposium on Information Technology: Coding and Computing*, pages 130–134, 2001.

[11] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless Ad-Hoc sensor networks. In *Proc. of MobiCOM*, pages 139–150, 2001.

[12] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage Problems in Wireless Ad-hoc Sensor Networks. In *Proc. of INFOCOM*, pages 1380–1387, 2001.

[13] S. Meguerdichian, S. Slijepcevic, V. Karayan and M. Potkonjak. Localized algorithms in wireless ad-hoc networks: location discovery and sensor exposure. In *Proc. of MobiHOC*, pages 106–116, 2001.